

kubernetes1.13安装

环境准备

-----环境信息-----
192.168.1.111 master
192.168.1.112 node1
192.168.1.113 node2

配置好主机名/etc/hosts (所有节点)

```
cat > /etc/test <<EOF
127.0.0.1    localhost
192.168.1.111 master
192.168.1.112 node1
192.168.1.113 node2
EOF
```

关闭SELinux和防火墙 (所有节点)

```
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
setenforce 0
systemctl stop firewalld && systemctl disable firewalld
```

关闭swap (所有节点)

```
swapoff -a
#限制开机启动
sed -i 's/\dev\mapper\centos-swap/#\dev\mapper\centos-swap/g' /etc/fstab
```

配置各节点系统内核参数使流

过网桥的流量也进入iptables/netfilter框架中 (所有节点)

#解决sysctl -p生效时报错 执行以下三句命令

yum install -y bridge-utils.x86_64

modprobe bridge

modprobe br_netfilter

#以上执行完再执行下面的指令 以免报错

```
cat <<EOF > /etc/sysctl.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
sysctl -p
```

配置阿里K8S YUM源 (所有节点)

```
cat > /etc/yum.repos.d/kubernetes.repo <<EOF
[kubernetes]
name=Kubernetes
baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=0
repo_gpgcheck=0
EOF
```

组件安装

docker安装 (所有节点)

```
yum -y install docker
systemctl enable docker && systemctl start docker
```

配置阿里镜像加速器 如果有私仓需要信任配置再启用第二句

```
sudo tee /etc/docker/daemon.json <<-'EOF'
{
  "registry-mirrors": ["https://h9dkpwpe.mirror.aliyuncs.com"],
  # "insecure-registries":["192.168.53.114"]
}
EOF
systemctl daemon-reload
systemctl restart docker
```

kubeadm安装（所有节点）

默认安装最新版本

```
#yum install -y kubelet kubeadm kubectl
```

此处指定版本13.3：

```
yum install -y kubelet-1.13.3-0 kubeadm-1.13.3-0 kubectl-1.13.3-0
```

kubelet设置开机启动 并添加配置解决报错问题（所有节点）

```
systemctl enable kubelet
```

```
vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

添加以下内容：

```
Environment="KUBELET_CGROUP_ARGS="--cgroup-driver=systemd --runtime-cgroups=/systemd/system.slice --kubelet-cgroups=/systemd/system.slice"
```

```
systemctl start kubelet
```

```
systemctl enable kubelet
```

下载镜像

主节点镜像列表（master）

#查看所需的镜像及版本 附上可视化的镜像dashboard

#flannel镜像可以直接下，这里不作处理 这里主要解决google镜像无法下载问题

kubeadm config images list

```
[root@master ~]# kubeadm config images list
I0303 15:35:06.971044 39890 version.go:94] could not fetch a Kubernetes version from the internet: \
ps://storage.googleapis.com/kubernetes-release/release/stable-1.txt: net/http: request canceled while
ders)
I0303 15:35:06.972085 39890 version.go:95] falling back to the local client version: v1.13.3
k8s.gcr.io/kube-apiserver:v1.13.3
k8s.gcr.io/kube-controller-manager:v1.13.3
k8s.gcr.io/kube-scheduler:v1.13.3
k8s.gcr.io/kube-proxy:v1.13.3
k8s.gcr.io/pause:3.1
k8s.gcr.io/etcd:3.2.24
k8s.gcr.io/coredns:1.2.6
[root@master ~]#
```

```
docker pull mirrorgooglecontainers/kube-apiserver-amd64:v1.13.3
```

```
docker pull mirrorgooglecontainers/kube-controller-manager-amd64:v1.13.3
```

```
docker pull mirrorgooglecontainers/kube-scheduler-amd64:v1.13.3
```

```
docker pull mirrorgooglecontainers/kube-proxy-amd64:v1.13.3
```

```
docker pull mirrorgooglecontainers/pause-amd64:3.1
```

```
docker pull mirrorgooglecontainers/etcd-amd64:3.2.24
```

```
docker pull mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.1
```

```
docker pull coredns/coredns:1.2.6
```

或者：carlziess/coredns-1.2.6

```
docker tag mirrorgooglecontainers/kube-apiserver-amd64:v1.13.3 k8s.gcr.io/kube-apiserver:v1.13.3
docker tag mirrorgooglecontainers/kube-controller-manager-amd64:v1.13.3 k8s.gcr.io/kube-controller-manager:v1.13.3
docker tag mirrorgooglecontainers/kube-scheduler-amd64:v1.13.3 k8s.gcr.io/kube-scheduler:v1.13.3
docker tag mirrorgooglecontainers/kube-proxy-amd64:v1.13.3 k8s.gcr.io/kube-proxy:v1.13.3
docker tag mirrorgooglecontainers/pause-amd64:3.1 k8s.gcr.io/pause:3.1
docker tag mirrorgooglecontainers/etcd-amd64:3.2.24 k8s.gcr.io/etcd:3.2.24
docker tag coredns/coredns:1.2.6 k8s.gcr.io/coredns:1.2.6
docker tag mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.1 k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1
```

运行节点镜像列表 (node1、node2)

#flannel镜像可以直接下，这里不作处理 这里主要解决google镜像无法下载问题

```
docker pull mirrorgooglecontainers/kube-proxy-amd64:v1.13.3
docker pull mirrorgooglecontainers/pause-amd64:3.1
```

```
docker tag mirrorgooglecontainers/kube-proxy-amd64:v1.13.3 k8s.gcr.io/kube-proxy:v1.13.3
docker tag mirrorgooglecontainers/pause-amd64:3.1 k8s.gcr.io/pause:3.1
```

主节点初始化

```
kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=0.0.0.0
```

```
1
2 [root@master ~]# kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=0.0.0.0
3 [init] Using Kubernetes version: v1.13.3
4 [preflight] Running pre-flight checks
5 error execution phase preflight: [preflight] Some fatal errors occurred:
6 [ERROR Swap]: running with swap on is not supported. Please disable swap
7 [preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
8 [root@master ~]# swapoff -a
9 [root@master ~]# #限制开机启动
10 [root@master ~]# sed -i 's/\dev/mapper/centos-swap/#\dev/mapper/centos-swap/g' /etc/fstab
11 [root@master ~]# kubeadm init --kubernetes-version=v1.13.3 --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=0.0.0.0
12 [init] Using Kubernetes version: v1.13.3
13 [preflight] Running pre-flight checks
14 [preflight] Pulling images required for setting up a Kubernetes cluster
15 [preflight] This might take a minute or two, depending on the speed of your internet connection
16 [preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
17 [kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
18 [kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
19 [kubelet-start] Activating the kubelet service
20 [certs] Using certificateDir folder "/etc/kubernetes/pki"
21 [certs] Generating "etcd/ca" certificate and key
22 [certs] Generating "etcd/peer" certificate and key
23 [certs] etcd/peer serving cert is signed for DNS names [master localhost] and IPs [192.168.1.111 127.0.0.1 ::1]
24 [certs] Generating "etcd/healthcheck-client" certificate and key
25 [certs] Generating "etcd/server" certificate and key
26 [certs] etcd/server serving cert is signed for DNS names [master localhost] and IPs [192.168.1.111 127.0.0.1 ::1]
27 [certs] Generating "apiserver-etcd-client" certificate and key
28 [certs] Generating "ca" certificate and key
29 [certs] Generating "apiserver-kubelet-client" certificate and key
30 [certs] Generating "apiserver" certificate and key
31 [certs] apiserver serving cert is signed for DNS names [master kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.1.111]
32 [certs] Generating "front-proxy-ca" certificate and key
```

```

33 [certs] Generating "front-proxy-client" certificate and key
34 [certs] Generating "sa" key and public key
35 [kubeconfig] Using kubeconfig folder "/etc/kubernetes"
36 [kubeconfig] Writing "admin.conf" kubeconfig file
37 [kubeconfig] Writing "kubelet.conf" kubeconfig file
38 [kubeconfig] Writing "controller-manager.conf" kubeconfig file
39 [kubeconfig] Writing "scheduler.conf" kubeconfig file
40 [control-plane] Using manifest folder "/etc/kubernetes/manifests"
41 [control-plane] Creating static Pod manifest for "kube-apiserver"
42 [control-plane] Creating static Pod manifest for "kube-controller-manager"
43 [control-plane] Creating static Pod manifest for "kube-scheduler"
44 [etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
45 [wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s
46 [apiclient] All control plane components are healthy after 22.004987 seconds
47 [uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
48 [kubelet] Creating a ConfigMap "kubelet-config-1.13" in namespace kube-system with the configuration for the kubelets in the cluster
49 [patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "master" as an annotation
50 [mark-control-plane] Marking the node master as control-plane by adding the label "node-role.kubernetes.io/master=/"
51 [mark-control-plane] Marking the node master as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]
52 [bootstrap-token] Using token: j9fg3l.0fxvkxmxgtjyrgxn
53 [bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
54 [bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
55 [bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
56 [bootstraptoken] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
57 [bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
58 [addons] Applied essential addon: CoreDNS
59 [addons] Applied essential addon: kube-proxy
60
61 Your Kubernetes master has initialized successfully!
62
63 To start using your cluster, you need to run the following as a regular user:
64
65 mkdir -p $HOME/.kube
66 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
67 sudo chown $(id -u):$(id -g) $HOME/.kube/config
68
69 You should now deploy a pod network to the cluster.
70 Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
71 https://kubernetes.io/docs/concepts/cluster-administration/addons/
72
73 You can now join any number of machines by running the following on each node
74 as root:
75
76 kubeadm join 192.168.1.111:6443 --token j9fg3l.0fxvkxmxgtjyrgxn --discovery-token-ca-cert-hash sha256:f2809ca9baf77d868444112a3ec1b28a0bcf5bc18e55f1fa5a6aefe72c170e8d
77
78 [root@master ~]#
79

```

执行提示内容：

MASTER节点

mkdir -p \$HOME/.kube

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

NODE节点

```
kubeadm join 192.168.1.111:6443 --token j9fg3l.0fxvkxmxgtjyrgxn --discovery-token-ca-cert-hash sha256:f2809ca9baf77d868444112a3ec1b28a0bcf5bc18e55f1fa5a6aefe72c170e8d
```

查看节点信息 目前NotReady是因为fl网络未配的原因，这里显示这样就对了

```
[root@master ~]# kubectl get node
```

NAME	STATUS	ROLES	AGE	VERSION
------	--------	-------	-----	---------

master	NotReady	master	3m	v1.13.3
--------	----------	--------	----	---------

node1	NotReady	<none>	47s	v1.13.3
-------	----------	--------	-----	---------

node2	NotReady	<none>	8s	v1.13.3
-------	----------	--------	----	---------

```
[root@master ~]#
```

安装flannel网络（主节点）

查看网络组件的版本：

<https://github.com/coreos/flannel/blob/master/Documentation/kube-flannel.yml> 最新版为11注意最后一句话修改版本号就可以apply上

```
1 mkdir -p /etc/cni/net.d/
2
3 cat <<EOF> /etc/cni/net.d/10-flannel.conf
4 {
5   "name": "cbr0",
6   "type": "flannel",
7   "delegate": {
8     "isDefaultGateway": true
9   }
10 }
11
12 EOF
13
14 mkdir /usr/share/oci-umount/oci-umount.d -p
15
16 mkdir /run/flannel/
17
18 cat <<EOF> /run/flannel/subnet.env
19 FLANNEL_NETWORK=10.244.0.0/16
20 FLANNEL_SUBNET=10.244.1.0/24
21 FLANNEL_MTU=1450
22 FLANNEL_IPMASQ=true
23
24 EOF
25
26 kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/v0.11.0/Documentation/kube-flannel.yml
27
```

这个过程会比较慢，因为需要去现pull flannel的 docker镜像

```
kubectl get all -n kube-system
```

可以看到正在初始化

```
[root@master ~]# kubectl get all -n kube-system
NAME                READY   STATUS    RESTARTS   AGE
pod/coredns-86c58d9df4-7hkh  0/1     Pending   0           17m
pod/coredns-86c58d9df4-k4zj5  0/1     Pending   0           17m
pod/etcd-master       1/1     Running   0           16m
pod/kube-apiserver-master  1/1     Running   0           16m
pod/kube-controller-manager-master  1/1     Running   0           16m
pod/kube-flannel-ds-amd64-62nrr  0/1     Init:0/1   0           57s
pod/kube-flannel-ds-amd64-fprrq  0/1     Init:0/1   0           57s
pod/kube-flannel-ds-amd64-h2psq  0/1     Init:0/1   0           57s
pod/kube-proxy-5tawv     1/1     Running   0           13m
pod/kube-proxy-jf3m      1/1     Running   0           14m
pod/kube-proxy-x4th9     1/1     Running   0           17m
pod/kube-scheduler-master  1/1     Running   0           16m

NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
service/kube-dns     ClusterIP           10.96.0.10      <none>            53/UDP,53/TCP    17m

NAME                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/kube-flannel-ds-amd64  3         3         0       0             0           beta.kubernetes.io/arch=amd64  57s
daemonset.apps/kube-flannel-ds-arm    0         0         0       0             0           beta.kubernetes.io/arch=arm    57s
daemonset.apps/kube-flannel-ds-arm64  0         0         0       0             0           beta.kubernetes.io/arch=arm64  57s
daemonset.apps/kube-flannel-ds-ppc64le 0         0         0       0             0           beta.kubernetes.io/arch=ppc64le 57s
daemonset.apps/kube-flannel-ds-s390x   0         0         0       0             0           beta.kubernetes.io/arch=s390x  57s
daemonset.apps/kube-proxy             3         3         3       3             3           <none>                  17m

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/coredns  0/2     2             0           17m

NAME                DESIRED   CURRENT   READY   AGE
replicaset.apps/coredns-86c58d9df4  2         2         0       17m
[root@master ~]# ^C
[root@master ~]#
```

一段时间后 只有一个节点Ready

```
1 [root@master ~]# kubectl get node
2 NAME STATUS ROLES AGE VERSION
3 master NotReady master 19m v1.13.3
4 node1 Ready <none> 17m v1.13.3
5 node2 NotReady <none> 16m v1.13.3
```

查看正在初始化的pod信息，可以看到还在pull镜像

kubectl describe pod/kube-flannel-ds-amd64-62nrr -n kube-system

```
/var/run/secrets/kubernetes.io/serviceaccount from flannel-token-6v24m (ro)
Conditions:
  Type             Status
  Initialized       False
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  run:
    Type:          HostPath (bare host directory volume)
    Path:           /run
    HostPathType:
  cni:
    Type:          HostPath (bare host directory volume)
    Path:           /etc/cni/net.d
    HostPathType:
  flannel-cfg:
    Type:          ConfigMap (a volume populated by a ConfigMap)
    Name:          kube-flannel-cfg
    Optional:      false
  flannel-token-6v24m:
    Type:          Secret (a volume populated by a Secret)
    SecretName:    flannel-token-6v24m
    Optional:      false
QoS Class:       Guaranteed
Node-Selectors:  beta.kubernetes.io/arch=amd64
Tolerations:     :NoSchedule
                 node.kubernetes.io/disk-pressure:NoSchedule
                 node.kubernetes.io/memory-pressure:NoSchedule
                 node.kubernetes.io/network-unavailable:NoSchedule
                 node.kubernetes.io/not-ready:NoExecute
                 node.kubernetes.io/unreachable:NoExecute
                 node.kubernetes.io/unschedulable:NoSchedule
Events:
  Type Reason Age From Message
  ---
  Normal Scheduled 4m51s default-scheduler Successfully assigned kube-system/kube-flannel-ds-amd64-62nrr to node2
  Normal Pulling 4m50s kubelet, node2 pulling image "quay.io/coreos/flannel:v0.10.0-amd64"
[root@master ~]#
```

过了三五分钟后，全部Ready状态了

```
1 [root@master ~]# kubectl get node
2 NAME STATUS ROLES AGE VERSION
3 master Ready master 22m v1.13.3
4 node1 Ready <none> 20m v1.13.3
5 node2 Ready <none> 19m v1.13.3
6 [root@master ~]#
7
8
9 #健康检查
10 [root@master ~]# kubectl get cs
11 NAME STATUS MESSAGE ERROR
12 scheduler Healthy ok
13 controller-manager Healthy ok
14 etcd-0 Healthy {"health": "true"}
15 [root@master ~]#
```

重启所有节点，看开机是否正常运行

```
1 [root@master ~]# kubectl get cs
2 NAME STATUS MESSAGE ERROR
```

```
3 scheduler Healthy ok
4 controller-manager Healthy ok
5 etcd-0 Healthy {"health": "true"}
6 [root@master ~]#
```

接下来我们安装可视化界面

Kubernetes-DashBoard用户界面

第一步：安装

第二步：建账号

第三步：获取管理密码

第四步：WEB登录管理

第一步：安装

[wget https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml](https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml)

修改内容：将以下国内不可达的镜像地址改为docker hub可达的地址

```
1 vi kubernetes-dashboard.yaml
```

/****

#文件尾部添加红色 开启node端的访问入口：

#这里的坑就是大小写和间距 不能有TAB全部用空格对好！、

修改第一处：

image: k8s.gcr.io/kubernetes-dashboard-amd64:v1.10.1

改为

image: **mirrorgooglecontainers/kubernetes-dashboard-amd64:v1.10.1**

修改第二处：

type: NodePort

ports:

- port: 443

targetPort: 8443

nodePort: 30001

****/

```
1 [root@master ~]# kubectl apply -f kubernetes-dashboard.yaml
2 secret/kubernetes-dashboard-certs created
3 serviceaccount/kubernetes-dashboard created
4 role.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
5 rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
6 deployment.apps/kubernetes-dashboard created
7 service/kubernetes-dashboard created
```

修改完文件内容如下附件



kubernetes-dashbo..d.yaml
4.51KB

第二步：创建管理员账号

创建授权账号登录：admin-user.yaml

```
1 [root@master2 k8s]# vi admin-k8s.yaml
2 apiVersion: v1
3 kind: ServiceAccount
```

```
1 [root@master ~]# kubectl apply -f admin-k8s.yaml
2 serviceaccount/admin-user created
3 clusterrolebinding.rbac.authorization.k8s.io/admin-user created
4 [root@master ~]#
```

```
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')
```

```

1 [root@master ~]# kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')
2 Name: admin-user-token-vwkbc
3 Namespace: kube-system
4 Labels: <none>
5 Annotations: kubernetes.io/service-account.name: admin-user
6   kubernetes.io/service-account.uid: 877d81bf-3d8f-11e9-bb35-000c298e17b8
7
8 Type: kubernetes.io/service-account-token
9
10 Data
11 ====
12 ca.crt: 1025 bytes
13 namespace: 11 bytes
14 token: eyJhbGciOiJIUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhbnVudW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWN1YWNBjB3VudC9uYyW1c3BhY2UiOiJrdWJlcm5ldGVzL3NlcnZpY2VhbnVudW50IiwiaWF0IjoiMTYyMjY2UTUyZWYwNmVudC51aWQ0Ii4NzkODFiZi0zZHhmLTExZTktyMIZNS0wMDBjbMjk4ZTE3YjgiLCJzdWIiOiJzeXN0ZW06c2Vydm1jZWZjY291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlcjI9.rQo5mLr1Akxr-glpgKySpR4h1ieZE2w4SGmucbCqP1vK2osR6R434_dNgp3ML1_v02FBysTZbPoKa8FnIxOkun2wzNfhmwZBuwcFlI2oJzUUvUbmvGmfmdrVESQqpMvQ9b392nE7CXyuLYnzNVNxJWETn5MH2JxyVLrzsw8eSO_RMcWQA0xgfGqSHuCc7aFgwcSpbd50nmb2wbXg0ji2fz9CRvtDzakPMHop-vXR6JngopxlEyTfALbjL-gGyUr00501uoXfzhYGJzcWxELYI9PWcbewSZkr6oyriLkRzZVy0PZe0e179ThhnuO6rmly9-eVM1o2LOBusPS8jbW

```

```
1 [root@master ~]# kubectl get svc -n kube-system -o wide
2 NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR
3 kube-dns ClusterIP 10.96.0.10 <none> 53/UDP,53/TCP 53m k8s-app=kube-dns
4 kubernetes-dashboard NodePort 10.106.93.53 <none> 443:30001/TCP 10m k8s-app=kubernetes-dashboard
5
6 [root@master ~]# kubectl get pods -n kube-system -o wide
7 NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
8 kubernetes-dashboard-76479d66bb-hk2pb 1/1 Running 0 11m 10.244.2.2 node2 <none> <none>
```

查得IP为：10.106.93.53 运行于node2

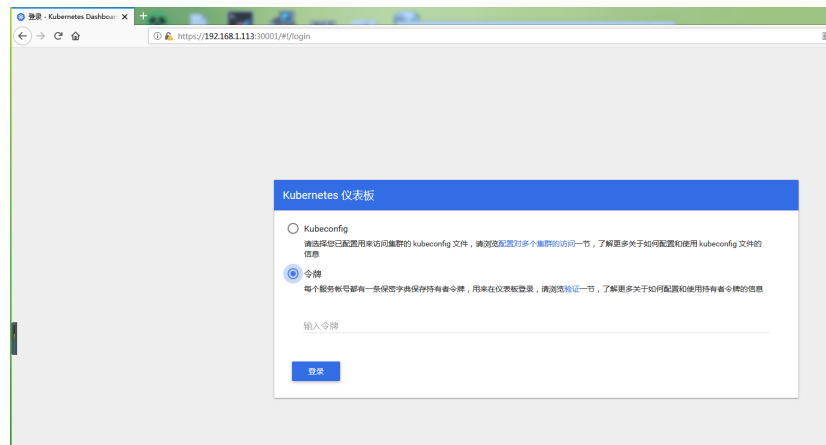
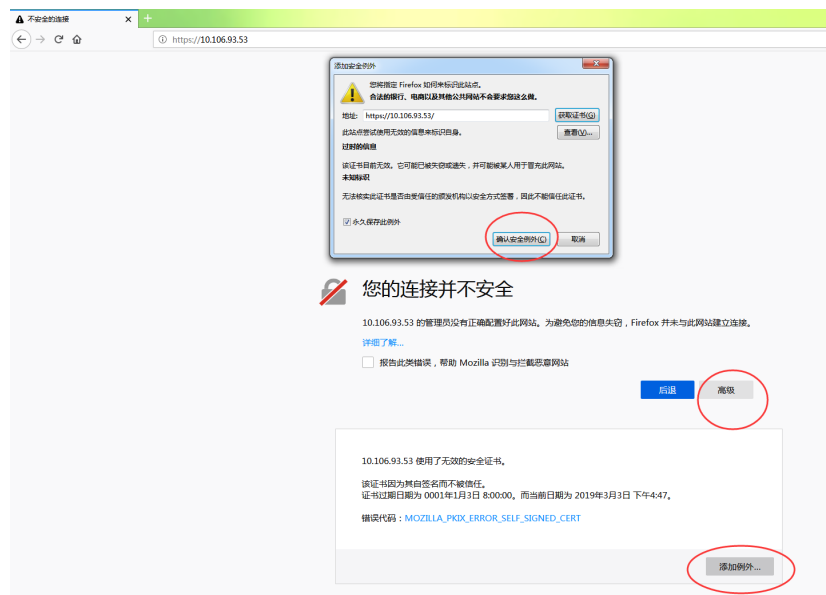
添加路由访问：<https://10.106.93.53>

或者用NodePort访问：<https://192.168.1.113:30001>

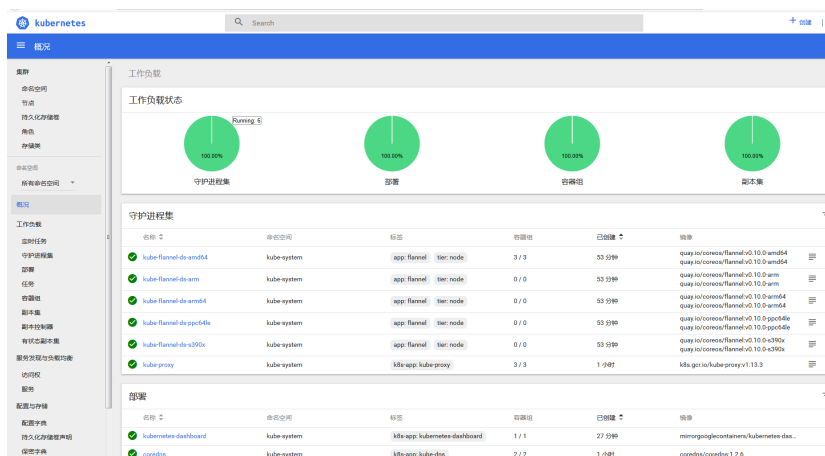
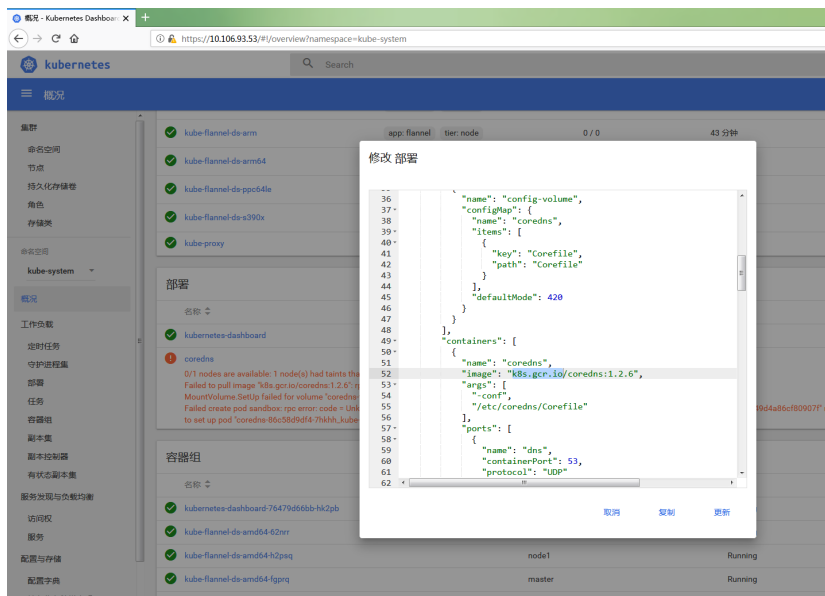
注意点：此处最好用火狐 添加不安全网站例外后可以正常访问 密码就是上面的token

添加路由

```
1 Microsoft Windows [版本 6.1.7601]
2 版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
3
4 C:\Users\Administrator>route add 10.106.0.0 mask 255.255.0.0 192.168.1.111
5 操作完成!
```



登录后发现还有容器因为翻墙问题出错，改掉它为：coredns/coredns:1.2.6 更新后一切完美



解决DNS问题

干掉core-dns

kubectl delete deployment.apps/coredns -n kube-system

kubectl delete svc coredns -n kube-system

kubectl delete svc kube-dns -n kube-system

官网文件：<https://github.com/kubernetes/kubernetes/blob/release-1.8/cluster/addons/dns/kubedns-controller.yaml.base>

wget <https://github.com/kubernetes/kubernetes/blob/release-1.8/cluster/addons/dns/kubedns-controller.yaml.base>

mv kubedns-controller.yaml.base kube-dns.yaml

里面修改了域名和IP地址,以下这两步已完成替换在下面的yaml内容里

```
1 sed -i 's/___PILLAR_DNS_SERVER___/10.96.0.10/g' kube-dns.yaml
2 sed -i 's/___PILLAR_DNS_DOMAIN___/cluster.local/g' kube-dns.yaml
3
```

```
1 vi kube-dns.yaml
2 # Copyright 2016 The Kubernetes Authors.
3 #
4 # Licensed under the Apache License, Version 2.0 (the "License");
5 # you may not use this file except in compliance with the License.
6 # You may obtain a copy of the license at
7 #
8 # http://www.apache.org/licenses/LICENSE-2.0
```

```
9 #
10 # Unless required by applicable law or agreed to in writing, software
11 # distributed under the License is distributed on an "AS IS" BASIS,
12 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 # See the License for the specific language governing permissions and
14 # limitations under the License.
15
16 # Should keep target in cluster/addons/dns-horizontal-autoscaler/dns-horizontal-autoscaler.yaml
17 # in sync with this file.
18
19 # __MACHINE_GENERATED_WARNING__
20
21 apiVersion: v1
22 kind: Service
23 metadata:
24   name: kube-dns
25   namespace: kube-system
26   labels:
27     k8s-app: kube-dns
28     kubernetes.io/cluster-service: "true"
29     addonmanager.kubernetes.io/mode: Reconcile
30     kubernetes.io/name: "KubeDNS"
31 spec:
32   selector:
33     k8s-app: kube-dns
34   clusterIP: 10.96.0.10
35   ports:
36   - name: dns
37     port: 53
38     protocol: UDP
39   - name: dns-tcp
40     port: 53
41     protocol: TCP
42 ---
43 apiVersion: v1
44 kind: ServiceAccount
45 metadata:
46   name: kube-dns
47   namespace: kube-system
48   labels:
49     kubernetes.io/cluster-service: "true"
50     addonmanager.kubernetes.io/mode: Reconcile
51 ---
52 apiVersion: v1
53 kind: ConfigMap
54 metadata:
55   name: kube-dns
56   namespace: kube-system
57   labels:
58     addonmanager.kubernetes.io/mode: EnsureExists
59 ---
60 apiVersion: apps/v1
61 kind: Deployment
62 metadata:
63   name: kube-dns
64   namespace: kube-system
65   labels:
66     k8s-app: kube-dns
67     kubernetes.io/cluster-service: "true"
68     addonmanager.kubernetes.io/mode: Reconcile
```

```
69 spec:
70 # replicas: not specified here:
71 # 1. In order to make Addon Manager do not reconcile this replicas parameter.
72 # 2. Default is 1.
73 # 3. Will be tuned in real time if DNS horizontal auto-scaling is turned on.
74 strategy:
75 rollingUpdate:
76 maxSurge: 10%
77 maxUnavailable: 0
78 selector:
79 matchLabels:
80 k8s-app: kube-dns
81 template:
82 metadata:
83 labels:
84 k8s-app: kube-dns
85 annotations:
86 scheduler.alpha.kubernetes.io/critical-pod: ''
87 seccomp.security.alpha.kubernetes.io/pod: 'docker/default'
88 spec:
89 priorityClassName: system-cluster-critical
90 securityContext:
91 supplementalGroups: [ 65534 ]
92 fsGroup: 65534
93 tolerations:
94 - key: "CriticalAddonsOnly"
95 operator: "Exists"
96 volumes:
97 - name: kube-dns-config
98 configMap:
99 name: kube-dns
100 optional: true
101 containers:
102 - name: kubedns
103 image: yxmu2006/k8s-dns-kube-dns:1.14.13
104 resources:
105 # TODO: Set memory limits when we've profiled the container for large
106 # clusters, then set request = limit to keep this container in
107 # guaranteed class. Currently, this container falls into the
108 # "burstable" category so the kubelet doesn't backoff from restarting it.
109 limits:
110 memory: 170Mi
111 requests:
112 cpu: 100m
113 memory: 70Mi
114 livenessProbe:
115 httpGet:
116 path: /healthcheck/kubedns
117 port: 10054
118 scheme: HTTP
119 initialDelaySeconds: 60
120 timeoutSeconds: 5
121 successThreshold: 1
122 failureThreshold: 5
123 readinessProbe:
124 httpGet:
125 path: /readiness
126 port: 8081
127 scheme: HTTP
128 # we poll on pod startup for the Kubernetes master service and
```

```
129 # only setup the /readiness HTTP server once that's available.
130 initialDelaySeconds: 3
131 timeoutSeconds: 5
132 args:
133 - --domain=cluster.local.
134 - --dns-port=10053
135 - --config-dir=/kube-dns-config
136 - --v=2
137 env:
138 - name: PROMETHEUS_PORT
139   value: "10055"
140 ports:
141 - containerPort: 10053
142   name: dns-local
143   protocol: UDP
144 - containerPort: 10053
145   name: dns-tcp-local
146   protocol: TCP
147 - containerPort: 10055
148   name: metrics
149   protocol: TCP
150 volumeMounts:
151 - name: kube-dns-config
152   mountPath: /kube-dns-config
153 - name: dnsmasq
154 image: yxmu2006/k8s-dns-dnsmasq-nanny:1.14.13
155 livenessProbe:
156   httpGet:
157     path: /healthcheck/dnsmasq
158     port: 10054
159     scheme: HTTP
160   initialDelaySeconds: 60
161   timeoutSeconds: 5
162   successThreshold: 1
163   failureThreshold: 5
164   args:
165   - -v=2
166   - -logtostderr
167   - --configDir=/etc/k8s/dns/dnsmasq-nanny
168   - --restartDnsmasq=true
169   - --
170   - -k
171   - --cache-size=1000
172   - --no-negcache
173   - --dns-loop-detect
174   - --log-facility=-
175   - --server=/cluster.local/127.0.0.1#10053
176   - --server=/in-addr.arpa/127.0.0.1#10053
177   - --server=/ip6.arpa/127.0.0.1#10053
178 ports:
179 - containerPort: 53
180   name: dns
181   protocol: UDP
182 - containerPort: 53
183   name: dns-tcp
184   protocol: TCP
185 # see: https://github.com/kubernetes/kubernetes/issues/29055 for details
186 resources:
187   requests:
188     cpu: 150m
```

```

189 memory: 20Mi
190 volumeMounts:
191 - name: kube-dns-config
192 mountPath: /etc/k8s/dns/dnsmasq-nanny
193 - name: sidecar
194 image: yxmu2006/k8s-dns-sidecar:1.14.13
195 livenessProbe:
196 httpGet:
197 path: /metrics
198 port: 10054
199 scheme: HTTP
200 initialDelaySeconds: 60
201 timeoutSeconds: 5
202 successThreshold: 1
203 failureThreshold: 5
204 args:
205 - --v=2
206 - --logtostderr
207 - --probe=kubedns,127.0.0.1:10053,kubernetes.default.svc.cluster.local,5,SRV
208 - --probe=dnsmasq,127.0.0.1:53,kubernetes.default.svc.cluster.local,5,SRV
209 ports:
210 - containerPort: 10054
211 name: metrics
212 protocol: TCP
213 resources:
214 requests:
215 memory: 20Mi
216 cpu: 10m
217 dnsPolicy: Default # Don't use cluster DNS.
218 serviceAccountName: kube-dns

```

kubectl create -f kube-dns.yaml

修改启动参数配置文件指定 DNS

vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

Environment="KUBELET_DNS_ARGS=--cluster-dns=10.96.0.10 --cluster-domain=cluster.local"

```

# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating the KUBELET_KUBEADM_ARGS variable dynamically
EnvironmentFile=/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last resort. Preferably, the user should use
# the NodeRegistration.KubeletExtraArgs object in the configuration files instead. KUBELET_EXTRA_ARGS should be sourced from this file.
Environment="KUBELET_DNS_ARGS=--cluster-dns=10.96.0.10 --cluster-domain=cluster.local"
EnvironmentFile=/etc/sysconfig/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS $KUBELET_DNS_ARGS

```

busybox用于测试内容如下：

```

1
2 apiVersion: v1
3 kind: Service
4 metadata:
5   name: default-subdomain
6 spec:
7   selector:
8     name: busybox
9   clusterIP: None
10  ports:
11  - name: foo # Actually, no port is needed.
12    port: 1234

```

```
13   targetPort: 1234
14   ---
15   apiVersion: v1
16   kind: Pod
17   metadata:
18     name: busybox1
19     labels:
20       name: busybox
21   spec:
22     hostname: busybox-1
23     subdomain: default-subdomain
24     containers:
25     - image: busybox
26       command:
27       - sleep
28       - "3600"
29     name: busybox
```



kube-dns.yaml
6.06KB

kubectl create -f busybox.yaml

测试通过

```
1 [root@master2 ~]# kubectl exec -it busybox1 -- nslookup kubernetes
2 Server: 10.96.0.10
3 Address: 10.96.0.10:53
4
5 Non-authoritative answer:
6 Name: kubernetes.default.svc.cluster.local
7 Address: 10.96.0.1
8
```